

Periodicity in tilings

Emmanuel Jeandel Pascal Vanier

September 22, 2009

Abstract

Tilings and tiling systems are an abstract concept that arise both as a computational model and as a dynamical system. In this paper, we characterize the sets of periods that a tiling system can produce. We prove that up to a slight recoding, they correspond exactly to languages in the complexity classes **NSPACE**(n) and **NE**.

Keywords: Computational and structural complexity, tilings, dynamical systems.

Introduction

The model of tilings was introduced by Wang [16] in the 60s to study decision problems on some classes of logical formulas. Roughly speaking, we are given some local constraints (a *tiling system*), and we consider colorings of the plane that respect these constraints (a *tiling*).

This model has been discovered to be quite interesting both as a computation model [4] and as a two-dimensional analog of symbolic dynamics [14, 13]. In a sense, the results we obtained are an answer to a question of symbolic dynamics via a computational point-of-view.

In this paper we consider *periods* of tilings. A coloring of the plane is said to be periodic of period p if it is invariant by a translation by a (horizontal/vertical) distance of p . It is easy to build a tiling system so that there exists a tiling of period 1 but no tiling of any other period. It is also routine to build a tiling system for which there is no tiling, hence no periodic tiling. The first important result on the theory of tilings [4] states the existence of a tiling system which admits tilings, none of them being periodic. Our main result in this article is a characterization of the sets of periods we can obtain with tilings.

Periods in tilings are fundamental in the context of symbolic dynamics [14]. One of the main purposes of multidimensional symbolic dynamics is to decide whether two tiling systems generate roughly the same tilings or, more accurately, if the two *subshifts* they generate are conjugated. One of the main tools is conjugacy invariants, i.e. quantities that are preserved under conjugacy. The set of periods is such an invariant. One other well known example is the *entropy* which is a measure of the growth of the number of $n \times n$ squares that can appear inside a tiling.

It turns out that a characterization of entropy in tilings, itself a combinatorial quantity, is intimately linked with computational theory, more precisely recursivity: entropy of tilings are exactly the (non-negative) right recursively enumerable numbers [9]. This is merely an exception: most conjugacy invariants are to be expressed in terms of recursivity theory [1, 15].

We prove a similar theorem for periods in tilings, answering a open problem in [15]. For periodic tilings, which are truly *finite objects*, the good analog will not be *recursivity* but rather *complexity*. We will see indeed in this article that sets of tilings correspond to complexity classes, in particular to non-deterministic exponential space and non-deterministic exponential time. These classes are not unusual for a specialist in descriptive complexity [8]. This is not a coincidence; the reader familiar with this topic may indeed see the results in this article as an analog for tilings of the classical result of Jones and Selman [11] on spectra of first order formulas.

1 Tilings, periodicity and computations

Usually when considering tiling systems, Wang rules are used. Wang rules consider adjacent tiles only, whereas our set of rules will consider a neighborhood of tiles. This makes constructions easier to explain, while keeping almost the same properties as classical Wang tiling systems.

For any dimension $d \geq 1$, a tiling of \mathbb{Z}^d with a finite set of tiles T is a mapping $c : \mathbb{Z}^d \rightarrow T$. A tiling system is the pair (T, I) , where I is a finite set of forbidden patterns $I \subset T^N$, where $N \subset_{finite} \mathbb{Z}^d$ is the neighborhood. A tiling c is said *valid* if and only if none of the patterns of I ever appear in c . Since the number of forbidden patterns is finite, we could specify the rules by *allowed* patterns as well. We give an example of such a tiling system with the tiles of figure 1a and the forbidden patterns of figure 1b. The allowed tilings are showed in figure 2.

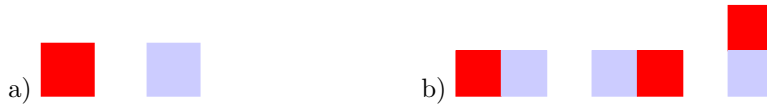


Figure 1: The set of tiles (a) and the forbidden patterns (b).

A tiling c of dimension $d = 2$ is said to be *horizontally periodic* if and only if there exists a *period* $p \in \mathbb{N}^*$ such that for all $x, y \in \mathbb{Z}$, $c(x, y) = c(x + p, y)$. A tiling c of \mathbb{Z}^d is *periodic* if it has the same period on all its dimensions:

$$c(x_1, x_2, \dots, x_d) = c(x_1 + p, x_2, \dots, x_d) = c(x_1, x_2, \dots, x_d + p)$$

A tiling system is *aperiodic* if and only if it tiles the plane but there is no valid periodic¹ tiling. Such tiling systems have been shown to exist with Wang

¹On none of the dimensions.

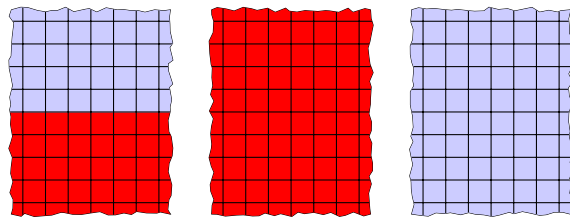


Figure 2: The only valid tilings of the system.

rules, and by extension for our set of rules. J. Kari and P. Papasoglu gave such a tiling system with an interesting property: determinism. A tiling system is *NW-deterministic* if given two diagonally adjacent tiles, there is only one way to complete the square in order to form a valid tiling. This means the two tiles *force* the next one, the mechanism is shown in figure 3.

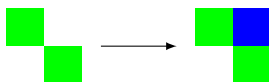


Figure 3: Determinism in the Kari-Papasoglu aperiodic tiling system.

Since the tiling system was described with Wang rules, it is easy to transpose it to our set of rules and change some details: instead of having the tiles to be adjacent diagonally, we can have them simply adjacent, as in figure 4. The type of determinism of this example will be called East-determinism. With such a tiling system, if a column of the plane is given, the half plane on its right is then determined.

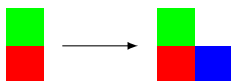


Figure 4: East-determinism in a tiling system.

2 Turing machines

As said earlier, tilings and recursivity are intimately linked. In fact, it is quite easy to encode Turing machines in tilings. We give in this section such an encoding. Similar encodings can be found e.g. in [12, 7].

Given a (non-deterministic, with one semi-infinite tape) Turing machine M , we build a tiling system τ_M in figure 5. The tiling system is given by Wang

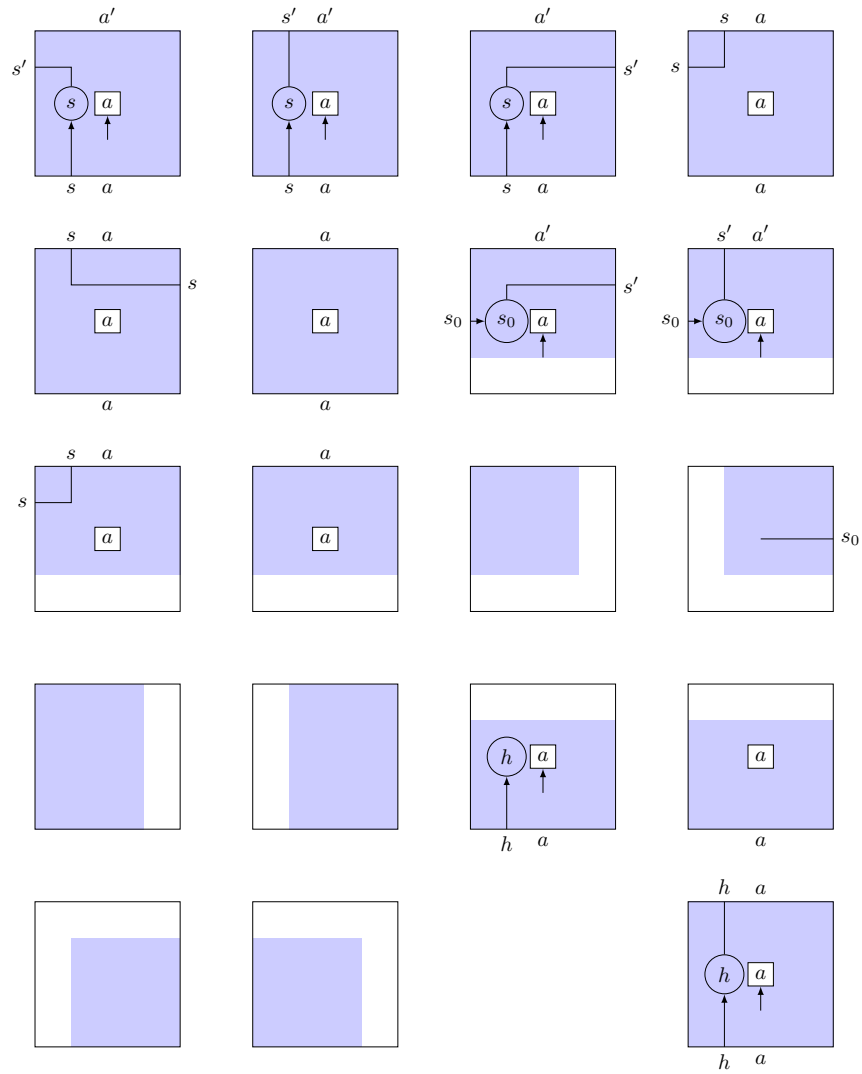


Figure 5: A tiling system, given by Wang tiles, simulating a Turing machine

tiles, i.e., we can only glue two tiles together if they coincide on their common edge. We now give some details on the picture:

- s_0 in the tiles is the initial state of the Turing machine.
- The first tile corresponds to the case where the Turing machine, given the state s and the letter a chose to go to the left and to change from s to s' , writing a' . The two other tiles are similar.
- h represents a halting state. Note that the only states that can appear in the last step of a computation (before a border appears) are halting states.

This tiling system τ_M has the following property: there is an accepting path for the word u in time (less than) w using space (less than) h if and only if we can tile a rectangle of size $(w + 2) \times h$ with white borders, the first row containing the input.

Note that this method works for both deterministic and non-deterministic machines. Using some usual tiling techniques, it is quite straightforward to build a tiling system encoding computation paths of n -tape Turing machine. As a consequence, complexity results stated below are also valid when the complexity classes are defined over n -tape Turing machines rather than one-tape Turing Machines.

3 Recognizing languages with tilings

We introduce here a way to recognize languages over \mathbb{N} , by looking at the set of periods of a given tiling system τ .

We define \mathcal{L}_τ^h , the language recognized by a tiling system τ , as the set of integers p such that there is a valid tiling of \mathbb{Z}^2 by τ of horizontal *eigenperiod* p . We mean by *eigenperiod* that p is the smallest non-zero period of the tiling, in order to avoid also having all multiples of any recognized number in the language. We similarly note \mathcal{L}_τ the set of eigenperiods of τ .

Definition 1. \mathcal{T} is the set of all languages \mathcal{L}_τ , where τ is any tiling system. \mathcal{H} is the set of all languages \mathcal{L}_τ^h , where τ is any tiling system. \mathcal{T} and \mathcal{H} are the classes of languages recognized by tiling and recognized horizontally by tiling respectively.

We say that a language \mathcal{L} is recognized by a tiling system τ if and only if $\mathcal{L} = \mathcal{L}_\tau^h$ or $\mathcal{L} = \mathcal{L}_\tau$, depending on the context.

A natural question that arises when studying languages concerns closure by union, intersection and complementation. Closure by union is easy to prove: take the disjoint union of the sets of tiles and forbid tiles from the first tiling system to be neighbors of tiles from the second one. Closure by intersection is not as straightforward, since the classical construction by cartesian product of the two tiling systems would lead to a language containing the *lcm* of the periods of each language.

In this paper, we prove the following:

Theorem 2. \mathcal{H} is closed by union, intersection and complementation.

Theorem 3. \mathcal{T} is closed by intersection. \mathcal{T} is closed by complementation if and only if $\mathbf{NE} = \mathbf{coNE}$.

Here \mathbf{NE} is the class of languages recognized by a (one-tape) non-deterministic Turing machine in time 2^{cn} for some $c > 0$. Note that for theorem 3 we need to work in any dimension d . That is, if $\mathcal{L}_1, \mathcal{L}_2$ are the set of periods of the tiling system τ_1, τ_2 , then there exists a tiling system τ' that corresponds to the set of period $\mathcal{L}_1 \cap \mathcal{L}_2$. However even if τ_1 and τ_2 are two-dimensional tiling systems, τ' might be in dimension greater than 2.

To prove these theorems, we will actually give a characterization in terms of structural complexity of our classes \mathcal{H} and \mathcal{T} . This will be the purpose of the next two sections. For an introduction to structural complexity, we suggest [2, 3].

4 \mathcal{H} and $\mathbf{NSPACE}(2^n)$

To formulate our theorem, we consider sets of periods, i.e. subsets of \mathbb{N}^* , as unary or binary languages. If $L \subset \mathbb{N}^*$ then we define $un(L) = \{1^{n-1} | n \in L\}$. We define $bin(L)$ to be the set of binary representations (missing the leading one) of numbers of L . As an example, if $L = \{1, 4, 9\}$, then $un(L) = \{\epsilon, 111, 11111111\}$ and $bin(L) = \{\epsilon, 00, 001\}$. Note that any language over the letter 1 (resp. the letters $\{0, 1\}$) is the unary (resp. binary) representation of some subset of \mathbb{N}^* .

We now proceed to the statement of the theorem:

Theorem 4. Let \mathcal{L} be a language, the following statements are equivalent:

- i) $\mathcal{L} \in \mathcal{H}$
- ii) $un(\mathcal{L}) \in \mathbf{NSPACE}(n)$
- iii) $bin(\mathcal{L}) \in \mathbf{NSPACE}(2^n)$

Recall that $\mathbf{NSPACE}(n)$ is the set of languages recognized by a (one-tape) non-deterministic Turing machine in space $O(n)$.

The $(ii) \Leftrightarrow (iii)$ is folklore from computational complexity theory. The following two lemmas will prove the equivalence $(ii) \Leftrightarrow (i)$ hence the result.

Lemma 5. For any tiling system τ , $un(\mathcal{L}_\tau^h) \in \mathbf{NSPACE}(n)$.

Proof. Let \mathcal{L}_τ^h be the language recognized by τ . We will construct a non-deterministic Turing machine accepting n if and only if it is a horizontal eigen-period of τ . The machine has to work in space $\mathcal{O}(n)$, the input being given in unary.

The tiling system has a finite number of finite rules, hence it has a "radius" of action r . A consequence of this is that if there exists a horizontally periodic tiling of period n , then there is such a tiling of height at most $|T|^{rn}$.

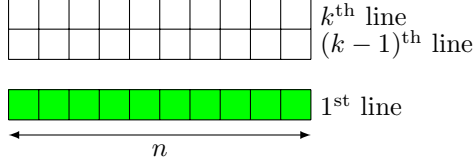


Figure 6: The tape of the machine when treating the k^{th} line: we keep the first line in memory to verify the validity of the last one, and we guess the next line non-deterministically.

This means that with a Turing machine working in space $n \times |T|^{rn}$, it is easy to verify if an integer n is a period of the tiling τ . We just guess non-deterministically a rectangle $n \times |T|^{rn}$ and check whether it is valid with copies of itself as neighbors. To check if it is an *eigenperiod* we just have to keep in memory a boolean for each integer smaller than n , and check for each of them if they are not a possible period.

Actually, we only have to remember r lines of the tiling at once to check the validity of some line. This allows us to bring the space back to $\mathcal{O}(n)$. The method is the same as stated before, we check non-deterministically each line, all the preceding $r - 1$ lines being stored to be able to check validity. We also keep the $r - 1$ first lines guessed, in order to check the validity of the last lines. Figure 6 shows the state of the tape while checking the k -th line for $r = 1$.

The machine hereby constructed works in space $\mathcal{O}(n)$, therefore $\mathcal{L}_\tau^h \in \mathbf{NSPACE}(2^n)$. \square

Lemma 6. *For any unary language $L \in \mathbf{NSPACE}(n)$, then $\{n \in \mathbb{N}^*, 1^{n-1} \in L\} \in \mathcal{H}$.*

Proof. Let $L \in \mathbf{NSPACE}(n)$. There exists then a non-deterministic Turing machine M accepting L in linear space. Using traditional tricks from complexity theory, we can suppose that, on input 1^n the Turing machine uses exactly $n + 1$ cells of the tape (i.e. the input, with one additional cell on the right) and works in time c^n for some constant c .

We will build a tiling system τ so that $1^n \in L$ if and only if $n + 4$ is a period of the tiling τ . The modification to obtain $n + 1$ rather than $n + 4$, and thus prove the lemma, is left to the reader (basically “fatten” the gray tiles presented below so that they absorb 3 adjacent tiles), and is of no interest rather than technical.

The tiling system will be made of several components, each of them having a specific goal. The components and their rules are as follows:

- The first component A is composed of an aperiodic E-deterministic tiling system, whose tiles will be called “whites”. We take the one from section 1. We add a “gray” tile. The rules forbid any pattern containing a white tile

above or below a gray tile. Hence a column containing a gray tile can only have gray tiles.

With this construction, an aperiodic tiling of period p will have gray columns at distance p of each other. The reason for that is that the white tile can only tile the plane aperiodically.

For the moment nothing forbids more than one gray column to appear inside a period. Figure 7 shows a possible form of a periodic tiling at this stage.

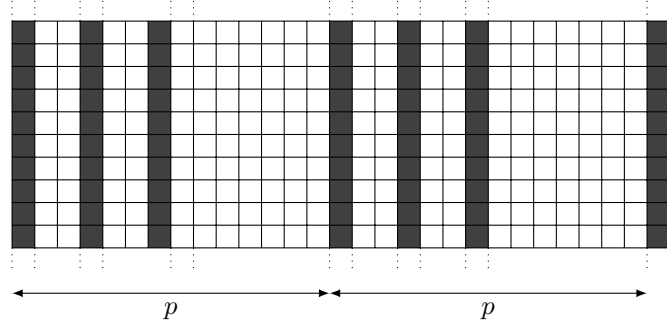


Figure 7: A periodic tiling with the tiling system A .

- The second component $D = P \times \{R, B\}$ is a tile set that will allow us to force the distance between gray columns to be always the same. Furthermore, if we call n this distance, then it will also force rows at distance c^n of each other. It is divided into two subcomponents P and $\{R, B\}$, which are respectively a counter in base c and a marker for the rows. Let us focus on the counter: it adds one on each row, which means, if the number is delimited by two gray columns at distance n , it can go from 0 to $c^n - 1$. Such a counter, can be realized with a transducer corresponding to the local operation.

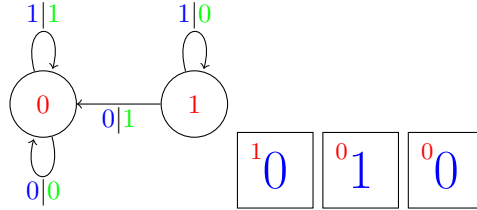


Figure 8: Transducer corresponding to the operation of adding a bit and its corresponding tiles. The states of the transducer correspond to the carry.

⁰ 1	¹ 0	¹ 0	
0	1	1	

Figure 9: Example of a valid tiling with the transducer tiles.

The transducer of figure 8 corresponds to the counter in the case $c = 2$.

To make a set of tiles out of this, we only need tiles containing the carry and the value at the same time. We forbid two tiles to be neighbors if the one on the left is not the result of applying the transducer to the one on the right. The gray columns being the delimiters, if a gray tile is on the right of some white tile, it is always considered as a tile containing a carry.

The rules on the $\{R, B\}$ subcomponent are easy: a tile on the right or the left of a R tile is always a R tile, the value inside R tiles can only be 0. A gray tile is R if the tile below it was a $c - 1$, that is to say when the number is reinitialized.

We made here rows of R distant of c^n , which in conjunction with the gray columns form regular rectangles on the plane.

Figure 10a shows some typical tiling at this stage: The aperiodic component can still be different between two gray columns, so the distance between these columns is not the period of the tiling.

- Component T is only a copy of A which allows us to synchronize the first white columns after the gray columns: synchronising these columns ensures that the aperiodic components between two gray columns are always the same, since the aperiodic tiles are E-deterministic. The rules are simple, two horizontal neighbors have the same value on this component and a tile having a gray tile on its left has the same value in A as in T .

At this stage, we have regular rectangles on all the plane, whose width correspond to the period of the tiling, as shown in figure 10b.

- The last component M is the component allowing us to encode Turing machines in each rectangle. We use the encoding τ_M we described previously in section 1. We force the computation to appear inside the white tiles : The white bottom borders must appear only in the row R , and the row below will have top borders. And the two tiles between the row R and the gray tiles are corner tiles. Finally, the input of the Turing machine (hence the row above the row R) consist of only “1” symbols, with a final blank symbol.

The Turing machines considered here being non-deterministic, there could be different valid transitions on two horizontally adjacent rectangles, that

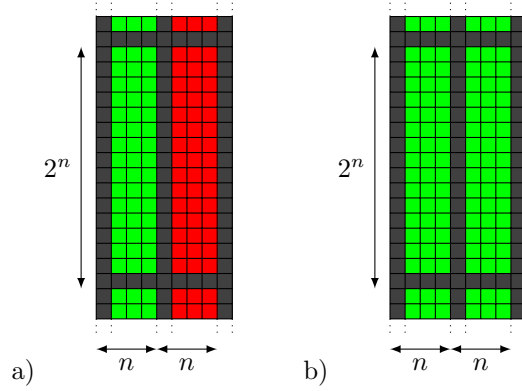


Figure 10: a) The form after adding the component D , the aperiodic components between two gray columns can be different. b) After adding also the component T , the aperiodic components are exactly the same.

is why we synchronize the transitions on each row. The method for the synchronization of the transition is almost the same as the method for the synchronisation of the aperiodic components, and thus not provided here.

Now we prove that $1^n \in L$ if and only if $n + 4$ is a period of the tiling system τ . First suppose that $n + 4$ is a period and consider a tiling of period $n + 4$.

- Due to component A , a gray column must appear. The period is a succession of either gray and white columns.
- Due to the component D , the gray columns are spaced by a period of p , $p < n + 4$.
- Due to component T the tiling we obtain is (horizontally) p -periodic when restricted to the components D, T, A .
- For the M component to be correctly tiled, the input 1^{p-4} ($4 = 1$ (gray) + 1 (left border) + 1 (right border) + 1 (blank marker)) must be accepted by the Turing Machine, hence $1^{p-4} \in L$
- Finally, due to the synchronization of the non-deterministic transition, the M component is also p -periodic. As a consequence, our tiling is p -periodic, hence $n + 4 = p$. Therefore $1^n \in L$

Conversely, suppose $1^n \in L$. Consider the coloring of period $n + 4$ obtained as follows (only a period is described):

- The component A consist of $n + 3$ correctly tiled columns of our aperiodic E -deterministic tiling systems, with an additional gray column. As the E -deterministic tiling system tiles the plane, such a tiling is possible

- The component M corresponds to a successful computation path of the Turing machine on the input 1^n , that exists by hypothesis.
- We then add all other layers according to the rules to obtain a valid configuration, thus obtaining a valid tiling of period exactly $n + 4$.

□

Corollary 7. *The languages recognized horizontally by tiling are closed under intersection and complementation.*

Proof. Immerman-Szelepczenyi's theorem [10, 3] states that non-deterministic space complexity classes are closed by complementation. The result is then a consequence of theorem 4. □

Note that if \mathcal{H} is closed under complementation, then $\mathbf{NSPACE}(2^n)$ is closed under complementation by theorem 4. As a consequence, any proof of corollary will translate into a proof that space complexity classes (greater than 2^n) are closed under complementation, hence give an alternate proof of Immerman-Szelepczenyi's result.

This theorem could be generalized to tilings of dimension d by considering tilings having a period n on $d - 1$ dimensions, as explained in [6].

5 \mathcal{T} and \mathbf{NE}

We now proceed to total periods rather than horizontal periods. We will prove:

Theorem 8. *Let $\mathcal{L} \subset \mathbb{N}^*$ be a language, the following statements are equivalent:*

- i) there exists a dimension d for which $\mathcal{L} \in \mathcal{T}$*
- ii) $un(\mathcal{L}) \in \mathbf{NP}$*
- iii) $bin(\mathcal{L}) \in \mathbf{NE}$*

We will obtain as a corollary Theorem 3. Note the slight difference in formulation between Theorem 3 and Theorem 4. While we can encode a Turing machine working in time n in a tiling of size n^2 , we cannot check the validity of the tiling in less than $\mathcal{O}(n^2)$ time steps. And furthermore, we cannot, as far as we know, check if n is an eigenperiod in time less than $\mathcal{O}(n^3)$. More generally, it is unclear that we can encode a Turing machine working in time n^c in a tiling of size less than n^{2c} . So instead of considering the time class $\mathbf{NTIME}(n)$, we consider the class \mathbf{NP} on unary inputs. This will allow us to cover the gap. The class \mathbf{NP} for unary inputs corresponds to the class \mathbf{NE} for binary inputs.

Proof. The statements $(i) \Rightarrow (ii) \Leftrightarrow (iii)$ were already explained. So we only have to prove $(ii) \Rightarrow (i)$.

In order to get a periodic structure from the construction of lemma 6, we only need to slightly modify it. We will detail the modification for $d = 2$, the generalization being straightforward:

- in component A, we start from an aperiodic NW-deterministic tiling system. We will consider 3 types of gray tiles: a crossing tile, a horizontal tile and a vertical tile. The rules are simple:
 - above or below a vertical tile one can only have a crossing or a vertical tile,
 - on the left or on the right of a horizontal tile, there can only be a crossing or a horizontal tile,
 - on the left and on the right of a crossing tile, there can only be a crossing or a horizontal tile,
 - above or below a crossing tile, there can only be a crossing or a vertical tile.

With these rules, when a row and a column of gray tiles cross, the crossing can only be a crossing tile.

- In component D, instead of using a counter, we only use a signal propagating in diagonal from each crossing tile, which can only end on a crossing tile. Instead of having rectangles of size $c^n \times n$ we now have squares of size $n \times n$. At this step in the proof, the reasoning is as follows: Take a periodic tiling, then either a horizontal or a vertical tile must appear, as the aperiodic tiling system cannot tile the plane periodically. Then by component D, an horizontal tile will force a vertical tile to appear, and vice versa, so that every periodic tilings is composed of squares.
- The component T will synchronize the aperiodic component between all squares, by propagating the first line and first column of a square to all the neighbors. Indeed, a square of a NW-deterministic tiling system is entirely fixed once we know the first line and the first column. This is easily done by synchronising as shown in figure 11.

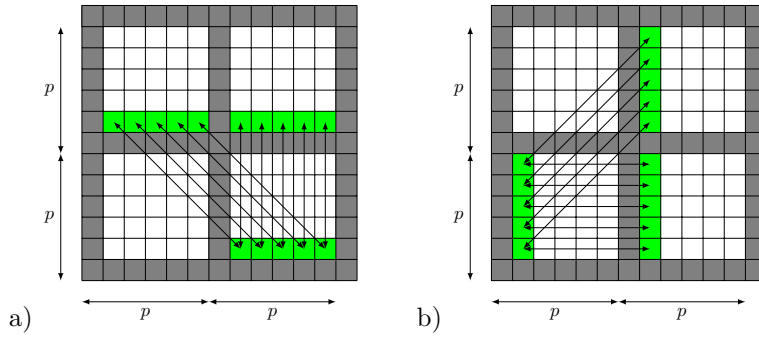


Figure 11: Transmission of the first row in a) and of the first line in b).

- The Turing Machine component is the same as before.

Now this proves that every unary language $L \in \mathbf{NTIME}(n)$ is the set of periods of a tiling system.

To prove the result for any unary language $L \in \mathbf{NP}$, we will need to work in higher dimensions : we will encode a machine working in time n^c in a tiling with $2c$ dimensions (c dimensions for the space and c dimensions for the time). The difficulty when generalising the dimension is not to keep the "hypercubical" structure, as the previously described structure can straightforwardly be transposed in any dimension d , but to keep the constraints of the Turing machine's tiles local. That is to say, two tape positions i and $i + 1$ have to be neighbors. The same goes for time, a content of a certain tape position at time t and its content at time $t + 1$ have to be neighbors.

Therefore, we will bend time and space... Actually, we will *fold* it as shown for dimension 3 in figure 12. Such a folding has already been described by

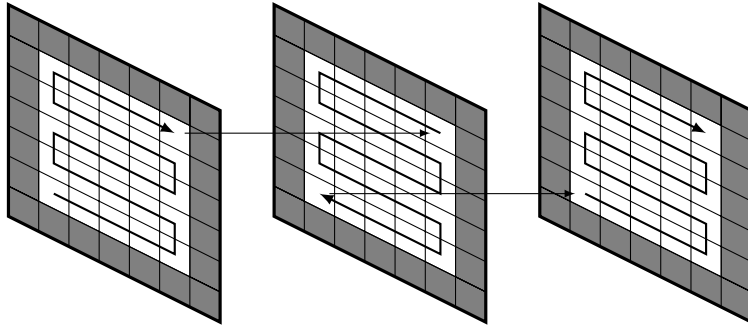


Figure 12: Folding of the tape in dimension 3.

Borchert [6] and can also be deduced from [11]. We do not detail the construction which is technical and rather tedious. The construction we obtain here allows us to code the Turing machine in a hypercube of dimension $2c$, hence proving the result. \square

Conclusion

We obtain in this paper a complete characterization of the sets of periods one can accomplish with tilings. The result is based on structural complexity theory, and is very different from the sets of periods we can obtain for 1-dimensional tilings (subshifts of finite type, see [14]) or for continuous discrete dynamical systems on the real line (Sharkovskii theorem, see e.g. [5]).

The next step is to characterize the *number* of tilings of period p for a given tiling system τ . Not surprisingly, preliminary work suggests we can characterize these functions via the class $\#P$.

References

- [1] Nathalie Aubrun and Mathieu Sablik. An order on sets of tilings corresponding to an order on languages. In *STACS*, pages 99–110, 2009.
- [2] J. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I*. Springer-Verlag, 1988.
- [3] J. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity II*. Springer-Verlag, 1988.
- [4] Robert Berger. *The Undecidability of the Domino Problem*. Number 66 in Memoirs of the American Mathematical Society. The American Mathematical Society, 1966.
- [5] Louis Block, John Guckenheimer, Michal Misiurewicz, and Lai Sang Young. Periodic points and topological entropy of one dimensional maps. In *Global Theory of Dynamical Systems*, pages 18–34.
- [6] Bernd Borchert. Formal Language characterizations of P, NP, and PSPACE. *Journal of Automata, Languages and Combinatorics*, 2009. to appear.
- [7] Gregory Chaitin. The Halting Probability via Wang Tiles. *Fundamenta Informaticae*, 86(4):429–433, 2008.
- [8] Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite Model Theory*. Springer Monographs in Mathematics. Springer, Berlin, 1995.
- [9] Michael Hochman and Tom Meyerovitch. A characterization of the entropies of multidimensional shifts of finite type. *Annals of Mathematics*, 2008.
- [10] Neil Immerman. Nondeterministic space is closed under complementation, 1988.
- [11] Neil D. Jones and Alan L. Selman. Turing machines and the spectra of first-order formulas with equality. In *STOC '72: Proceedings of the fourth annual ACM symposium on Theory of computing*, pages 157–167, New York, NY, USA, 1972. ACM.
- [12] Jarkko Kari. Reversibility and surjectivity problems of cellular automata. *J. Comput. Syst. Sci.*, 48(1):149–182, 1994.
- [13] Douglas Lind. Multidimensional Symbolic Dynamics. In *Symbolic dynamics and its applications*, volume 11 of *Proceedings of Symposia in Applied Mathematics*, pages 61–80, 2004.
- [14] Douglas A. Lind and Brian Marcus. *An Introduction to Symbolic Dynamics and Coding*. Cambridge University Press, New York, NY, USA, 1995.

- [15] Tom Meyerovitch. Growth-type invariants for \mathbb{Z}^d subshifts of finite type and classes arithmetical of real numbers. arXiv:0902.0223v1.
- [16] Hao Wang. Proving theorems by Pattern Recognition II. *Bell Systems technical journal*, 40:1–41, 1961.